vWEEKLY REPORT (7/12/23)-Krishna Tarun Saikonda

In this simulation, two types of vehicles are featured: 'default' and 'ego_vehicle,' each with unique attributes such as speed and acceleration. The file describes four different traffic flows for the default vehicles and specifies a unique 'ego_car' set to depart at a predetermined time in the simulation, indicating its importance in the research. The focus of this setup is on the interaction between normal traffic and the 'ego_vehicle' on a highway, providing an opportunity to observe diverse driving behaviors within a simulated urban setting for a highway environment.

Attributes of 'default' and 'ego_vehicle' types used in the simulation:

Parameter	Description	EGO VEHICLE	DEFAULT VEHICLE
carFollowModel	Model for following cars	EIDM	Krauss
sigma	Driver imperfection (0 = perfect, 1 = full imperfection)	0.5	0.5
tau	Time headway for following cars	0.01	0.3
accel	Maximum acceleration ability	6.0	4.0
sigmaerror	Error in the driver's estimation of its own speed	0.5	-
decel	Comfortable deceleration	10	5.0
emergencyDecel	Deceleration in emergency situations	5	5
maxSpeed	Maximum speed of the vehicle	45.0	35.0
collisionMinGapFactor	Factor for minimum gap in case of a collision	1	1
laneChangeModel	Model for changing lanes	SL2015	LC2013
lcAccelLat	Lateral acceleration during lane change	10	-
lcPushy	Aggressiveness in lane changing	1	-
lcImpatience	Impatience factor in lane changing	1	-
lcSigma	Driver imperfection in lane changing	1	-
lcSpeedGain	Speed gain during lane changing	10	-
lcTimeToImpatience	Time to impatience in lane changing	0.1	-
lcCooperative	Willingness to cooperate in lane changing	-1	-
jmSigmaMinor	Driver imperfection in minor junctions	1	-
jmIgnoreFoeSpeed	Speed threshold for ignoring foes in minor junctions	25	-
jmIgnoreFoeProb	Probability of ignoring foes in minor junctions	0.5	-
jmTimegapMinor	Time headway in minor junctions	0.3	-

Data Preprocessing:

The data obtained from the simulation undergoes further processing to make it suitable for training the model:

- XML Data Extraction: Data regarding the ego vehicle, all other vehicles, and collisions, known as Full-Car-Detector (FCD) data, is retrieved from XML files.
- Data Integration: The FCD data pertaining to the ego vehicle is combined with the collision data.
- Feature Development: New attributes are computed for the vehicles, which include:
  - Acceleration: The rate of change in velocity is calculated for each vehicle.
  - Closest Vehicle Identification and Proximity: At every time step for each vehicle, the nearest vehicle is identified by measuring the distance to all other vehicles. The proximity to these closest vehicles is also utilized as a characteristic.
  - Speed Differential: The speed difference between a vehicle and the vehicle closest to it is calculated.

The enhanced FCD data, now including these new attributes, is stored separately with specific emphasis on the ego vehicle's data, which will be used to train the model.

This procedure has been carried out for five different simulation runs, and the data from each run has been prepared as described above.

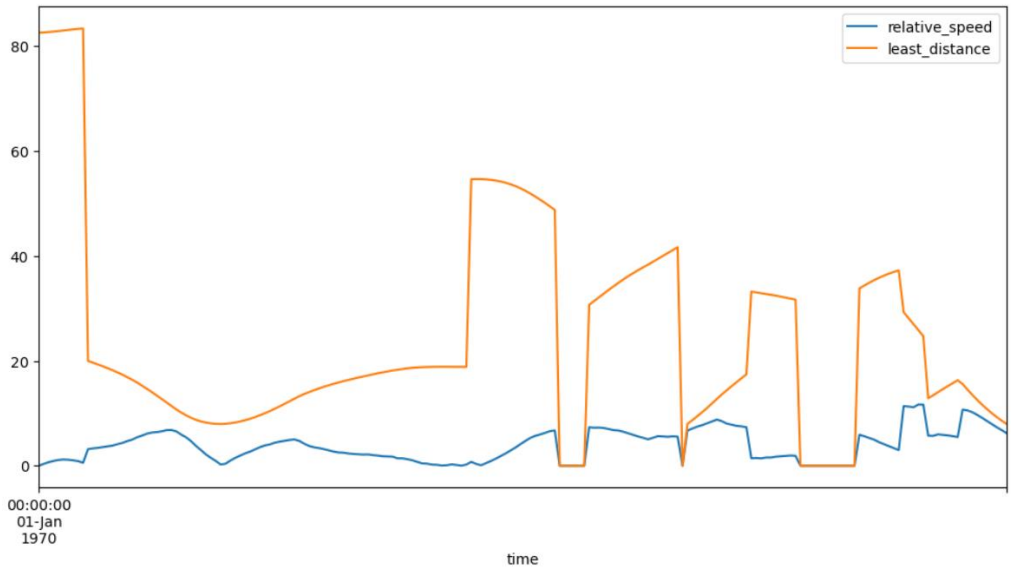
Dataset features (for all train-validation-test datasets):

Column	Description
time	Simulation time at which the data was recorded.
vehicle_id	Unique identifier for each vehicle.
x	X-coordinate of the vehicle's position.
y	Y-coordinate of the vehicle's position.
angle	Vehicle's orientation in degrees.
type	Type of the vehicle (e.g., EGOVEHICLE, default vehicle).
speed	Vehicle's speed at the recorded time.
position	Position of the vehicle on the road.
lane	Lane in which the vehicle is currently traveling.
slope	Slope of the road at the vehicle's current position.
acceleration	Vehicle's rate of change of speed (acceleration or deceleration).
nearest_vehicle_id	Identifier of the vehicle closest to this one.
least_distance	Distance to the nearest vehicle.
relative_speed	Difference in speed between this vehicle and its nearest vehicle.

I have executed four unique scenarios, employing two distinct datasets - one featuring individual data and the other a merged or combined dataset. To analyze these scenarios, I utilized two different predictive models: LSTM (Long Short-Term Memory) and ARIMA (Autoregressive Integrated Moving Average). This approach allowed me to extensively compare and contrast the performance and outcomes across a range of combinations, involving both individual and combined datasets, and applying each of the two different modeling techniques.

DATASETS:

SINGLE SIMULATION:



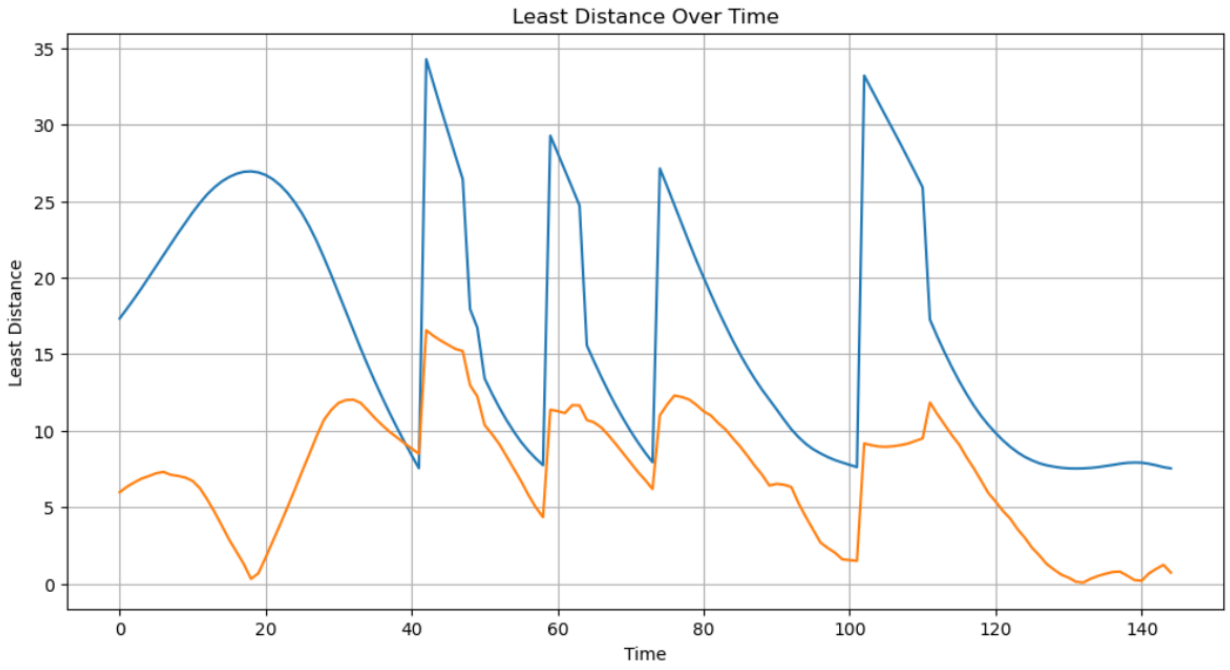
The 'relative_speed' trend, shown in blue, exhibits minor variations as it progresses over time. It maintains a relatively low and stable pattern, suggesting consistent speed differences between two entities, possibly vehicles, throughout the observed period.

The 'least_distance' trend, depicted in orange, is characterized by significant spikes and drops. It starts with a high value, drops sharply, and then fluctuates substantially, displaying sharp increases followed by decreases. This pattern suggests that the distance between the two entities changes dramatically over time, which could be interpreted as interactions such as overtaking or varying distances in traffic flow.

COMBINED DATA:

In the second Dataset, I concentrated on data from five different simulation runs. My focus was to refine this data by selecting only the information corresponding to the vehicle that was closest to the ego vehicle at the final time step recorded. This selection process was applied individually to each of the five simulation datasets. After filtering, I merged the datasets from each simulation to create a comprehensive dataset that specifically captured the dynamics between the ego vehicle and its nearest neighbor at the end of each simulation period. This consolidated dataset aimed to provide a more

focused analysis of the interactions between the ego vehicle and the single vehicle most proximate to it at the critical concluding moments of each simulation run.



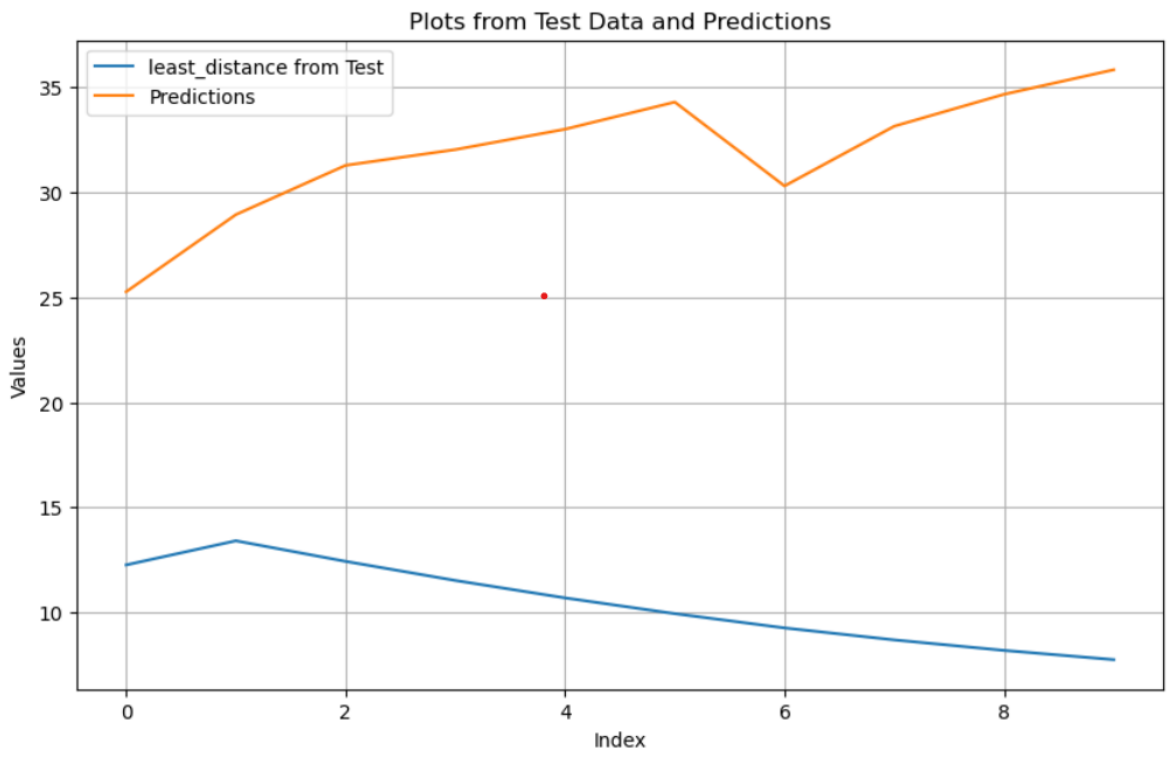
Scenario-1: Applying the LSTM Model on a single simulation data

LSTM MODEL: LSTM neural network using Keras, intended for sequence prediction tasks. The model consists of three LSTM layers with 150, 100, and 50 units, respectively, each using ReLU activation, with only the last LSTM layer returning the final output in the sequence. A Dropout layer with a rate of 0.2 follows to prevent overfitting by randomly omitting a portion of the layer's features during training. Then, two Dense layers are added; the first with 50 ReLU-activated units for non-linearity, and the second with 2 linear units for the final output prediction. The model employs the Adam optimizer and MSE loss function and is summarized to display the architecture and parameter count.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_6 (LSTM)               (None, 10, 150)           91800

 lstm_7 (LSTM)               (None, 10, 100)           100400

 lstm_8 (LSTM)               (None, 50)                30200

 dropout_2 (Dropout)         (None, 50)                0

 dense_4 (Dense)             (None, 50)                2550

 dense_5 (Dense)             (None, 2)                 102

=================================================================
Total params: 225052 (879.11 KB)
Trainable params: 225052 (879.11 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Results obtained after training the model :

Comparison of least_distance feature value predicted by the algorithm with the actual value - (least_distance is an essential feature for the collision of the vehicle)



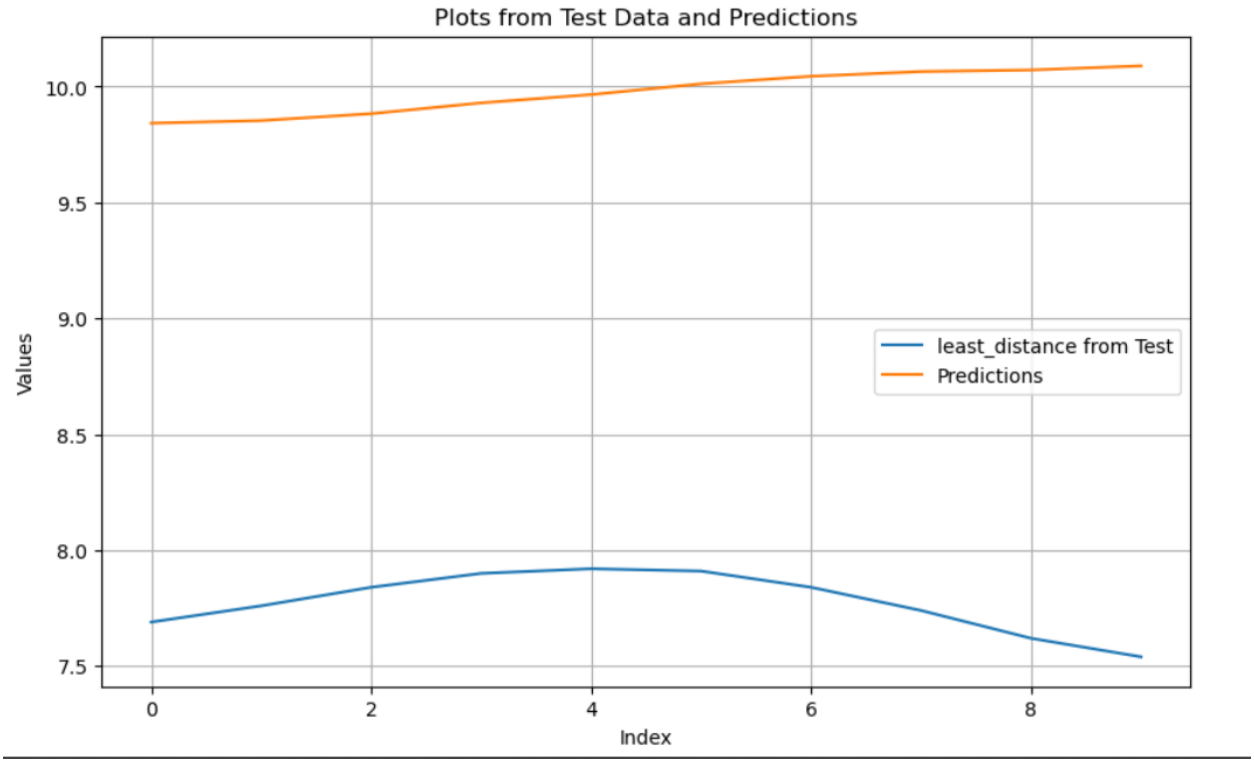
The graph presents a comparison between actual test data and predicted values, specifically for a metric referred to as 'least_distance'. Two lines are plotted: the blue line represents the actual 'least_distance' from the test data, and the orange line represents the predicted values.

From the graph, we can observe that the actual test data shows a decreasing trend over the indices, starting from just above 10 and descending towards a value close to 10 by the 8th index. On the other hand, the predicted values, shown by the orange line, increase over the indices, starting from a value just below 20 and reaching above 30 by the 8th index.

The significant divergence between the actual and predicted values suggests that the model may not be performing well at predicting the 'least_distance' metric accurately. The model seems to predict an increasing trend while the actual data is showing a decreasing trend. This could indicate that the model might need further tuning or training with more representative data to capture the underlying pattern more accurately. It's also possible that the model is not complex enough to capture the data's behavior or that there are other influencing factors not accounted for in the model.

Scenario-2: Applying the LSTM Model on Combined data

Results obtained after training the model:



The graph presents a comparison of two data sets: the original test data and a set of predicted values, presumably resulting from a model's performance. The original test data, depicted in blue, shows a trend where the 'least_distance' metric slightly increases, peaks around the midpoint of the index scale, and then decreases. The overall range of variation for the test data is between approximately 7.5 and a little over 8.5 on the y-axis.

In contrast, the predictions shown in orange maintain a flat line across the index, indicating that the model's predictions do not vary and stay constant at a value near 10, which does not follow the test data's trend.

The discrepancy between the test data and the predictions suggests that the model may not be capturing the variability present in the actual data, which could imply that the model needs further refinement to accurately predict the 'least_distance' metric's changes over time.

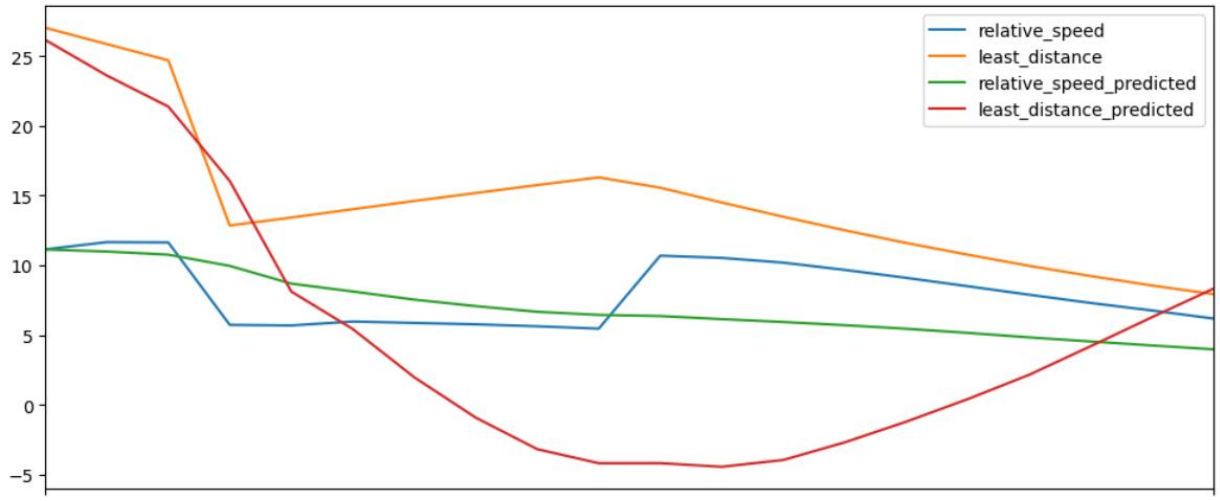
Scenario-3: Applying the ARIMA Model on a single simulation data

MODEL:

ARIMA stands for Autoregressive Integrated Moving Average, a popular statistical model used to forecast time series data. It combines three key aspects: autoregression (AR), which predicts future values based on past values; integration (I), which involves differencing the data to achieve stationarity; and moving average (MA), which models the error of the observation as a function of past errors. The model is parameterized by three values: p (the number of lag observations included in the model), d (the number

of times the data have been differenced), and q (the size of the moving average window). ARIMA is versatile and widely applied in various fields, from finance to meteorology, for its ability to model and predict data with trends and seasonalities.

Results obtained after training the model :



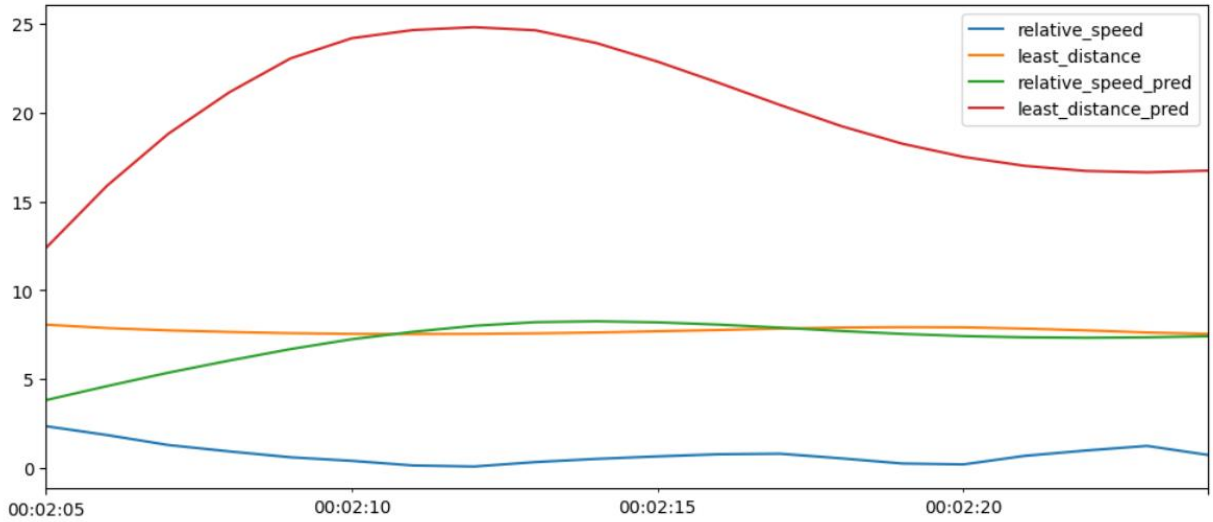
The graph displays the actual versus predicted data for two metrics: 'relative_speed' and 'least_distance'. The blue line represents the actual 'relative_speed', while the orange line depicts the actual 'least_distance'. The green and red lines correspond to the predicted 'relative_speed' and 'least_distance', respectively.

From the graph, it's apparent that the predicted 'relative_speed' does not closely follow the actual 'relative_speed', particularly in the first half of the plot where the actual values drop and then rise, whereas the predicted values remain relatively flat. Similarly, the predicted 'least_distance' (red line) starts off closely following the actual 'least_distance' but begins to diverge around the middle of the time series.

Both predicted lines seem to underestimate the values compared to the actual lines, with the discrepancy being more pronounced for the 'relative_speed'. Towards the end of the time series, while the actual 'least_distance' decreases, the predicted 'least_distance' increases, indicating a potential area where model accuracy could be improved. This suggests that the model may not be capturing the underlying patterns effectively for both 'relative_speed' and 'least_distance', and might benefit from further tuning.

Scenario-4: Applying the ARIMA Model on combined data

Results obtained after training the model :



The graph shows a time series comparison of actual and predicted values for two different metrics: 'relative_speed' and 'least_distance'. The x-axis, which represents time, is labeled with timestamps, suggesting that the data was collected over a short time interval.

The blue line represents the actual 'relative_speed', which increases over time, while the orange line represents the actual 'least_distance', which appears to peak and then decrease. The predictions for both metrics, shown in green ('relative_speed_pred') and red ('least_distance_pred'), follow a different trend compared to the actual data. The predicted 'relative_speed' remains relatively constant over the interval, while the predicted 'least_distance' shows a decrease followed by a leveling off.

The model's predictions for 'relative_speed' do not capture the increasing trend seen in the actual data, and while the 'least_distance' predictions capture a decreasing trend, they do not replicate the peak observed in the actual data. This suggests that the model may require further refinement to accurately model and predict the trends present in the actual data, particularly for capturing the nuances of the time series fluctuations.